Foundations of Stategic Business Analytics: Module 3: Predicting and Forecasting

Stefan Avey

2016-12-09

Contents

1 Credit Scoring Revisited

```
credit <- read.csv("DATA_3.01_CREDIT.csv")
credit2 <- read.csv("DATA_4.01_CREDIT2.csv")

dim(credit)
```

```
## [1] 300  10
```

```
dim(credit2)
```

```
## [1] 100  10
```

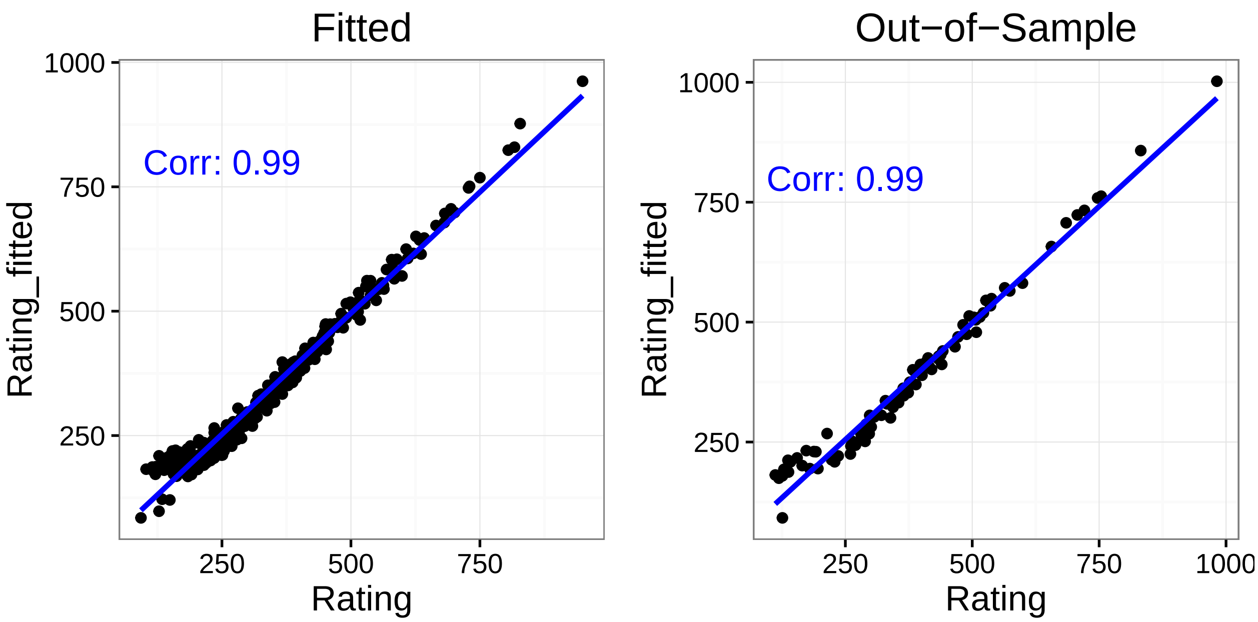
```
head(credit)
```

Income	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
14.891	283	2	34	11	Male	No	Yes	Caucasian	333

Income	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
106.025	483	3	82	15	Female	Yes	Yes	Asian	903
104.593	514	4	71	11	Male	No	No	Asian	580
148.924	681	3	36	11	Female	No	No	Asian	964
55.882	357	2	68	16	Male	No	Yes	Caucasian	331
80.180	569	4	77	10	Male	No	No	Caucasian	1151

1.1 Predict Credit Rating

```
## Build model to predict Credit Rating
lmod <- lm(Rating ~ ., credit)

credit <- mutate(credit, Rating_fitted = predict(lmod, newdata = credit))
credit2 <- mutate(credit2, Rating_fitted = predict(lmod, newdata = credit2))

## Original Dataset
lab <- paste0("Corr: ", round(cor(credit$Rating, credit$Rating_fitted), 2))
p1 <- ggplot(data = credit, aes(x = Rating, y = Rating_fitted)) +
  geom_point() +
  geom_smooth(color = "blue", method = lm, se = FALSE) +
  geom_text(data = data.frame(Rating = 250, Rating_fitted = 800),
            label = lab, color = "blue", size = 5) +
  ggtitle("Fitted") +
  getBaseTheme()

## New Dataset
lab <- paste0("Corr: ", round(cor(credit2$Rating, credit2$Rating_fitted), 2))
p2 <- ggplot(data = credit2, aes(x = Rating, y = Rating_fitted)) +
  geom_point() +
  geom_smooth(color = "blue", method = lm, se = FALSE) +
  geom_text(data = data.frame(Rating = 250, Rating_fitted = 800),
            label = lab, color = "blue", size = 5) +
  ggtitle("Out-of-Sample") +
  getBaseTheme()

Multiplot(p1, p2, cols = 2)
```



2 HR Analytics Revisited

```r
hr2 <- read.csv("DATA_3.02_HR2.csv")
hr3 <- read.csv("DATA_4.02_HR3.csv")

str(hr2)
```

```
## 'data.frame':    12000 obs. of  7 variables:
##  $ S      : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...
##  $ LPE    : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...
##  $ NP     : int  2 5 7 5 2 2 6 5 5 2 ...
##  $ ANH    : int  157 262 272 223 159 153 247 259 224 142 ...
##  $ TIC    : int  3 6 4 5 3 3 4 5 5 3 ...
##  $ Newborn: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left   : int  1 1 1 1 1 1 1 1 1 1 ...
```

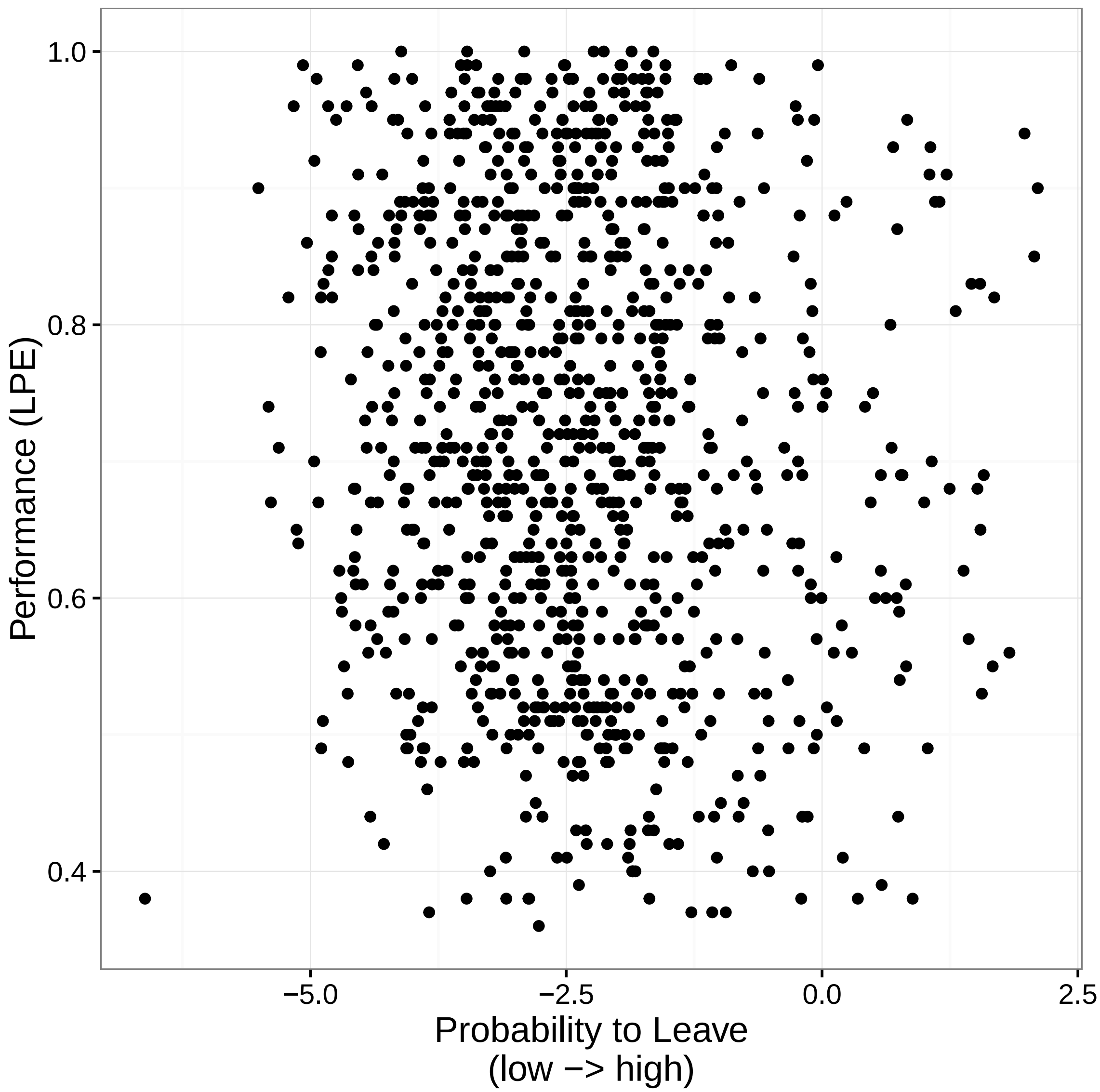
```r
str(hr3)
```

```
## 'data.frame':    1000 obs. of  6 variables:
##  $ S      : num  0.86 0.52 0.84 0.6 0.85 0.82 0.62 0.69 0.88 0.36 ...
##  $ LPE    : num  0.69 0.98 0.6 0.65 0.57 0.61 0.53 0.8 0.68 0.65 ...
##  $ NP     : int  4 4 5 3 3 4 3 3 5 5 ...
##  $ ANH    : int  105 209 207 143 227 246 128 219 236 119 ...
##  $ TIC    : int  4 2 2 2 2 3 4 3 3 5 ...
##  $ Newborn: int  1 0 0 1 0 0 0 1 0 0 ...
```

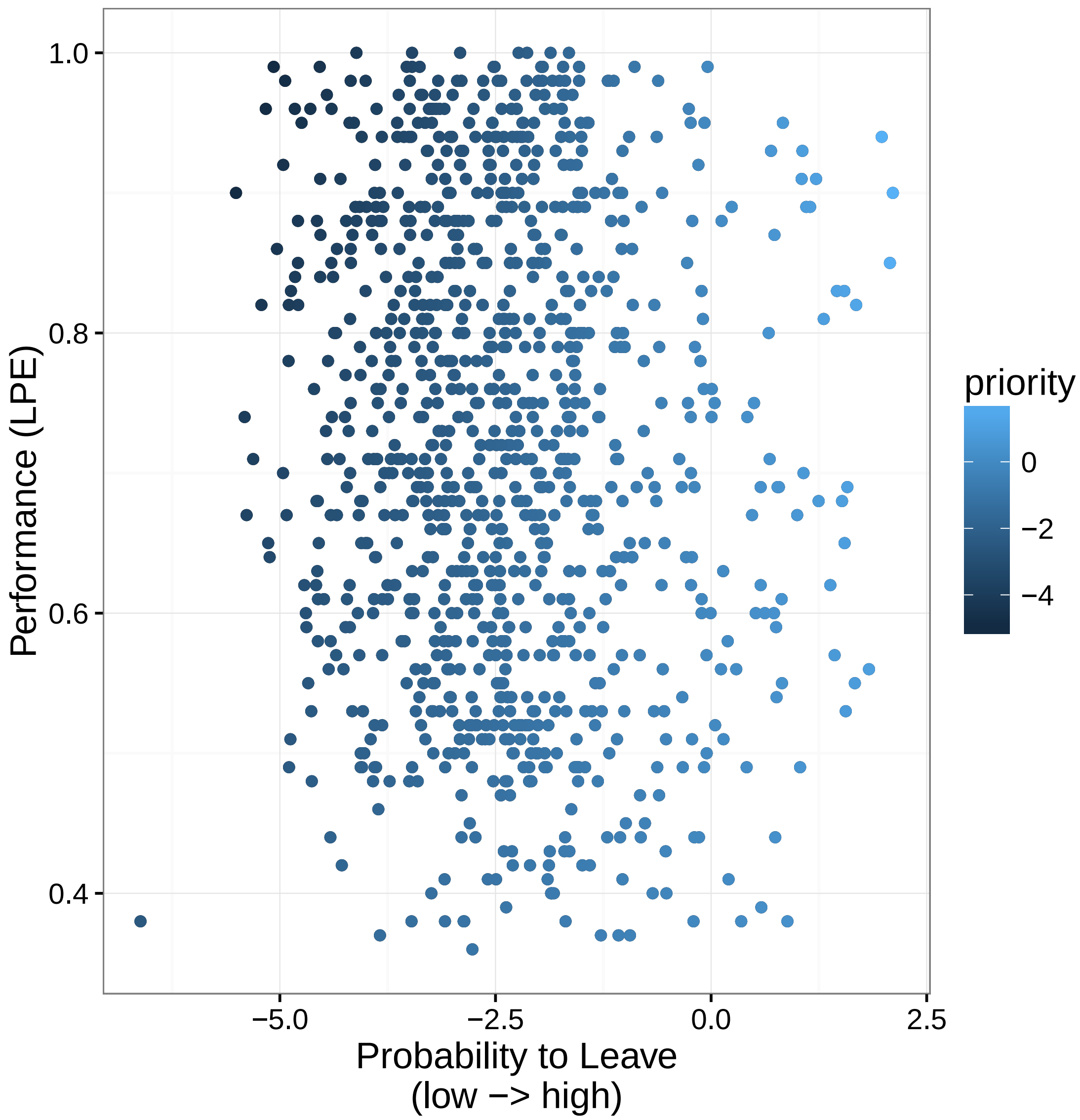
2.1 What Predicts Attrition?

We can use logistic regression to ask what factors will predict if an employee will leave the company.

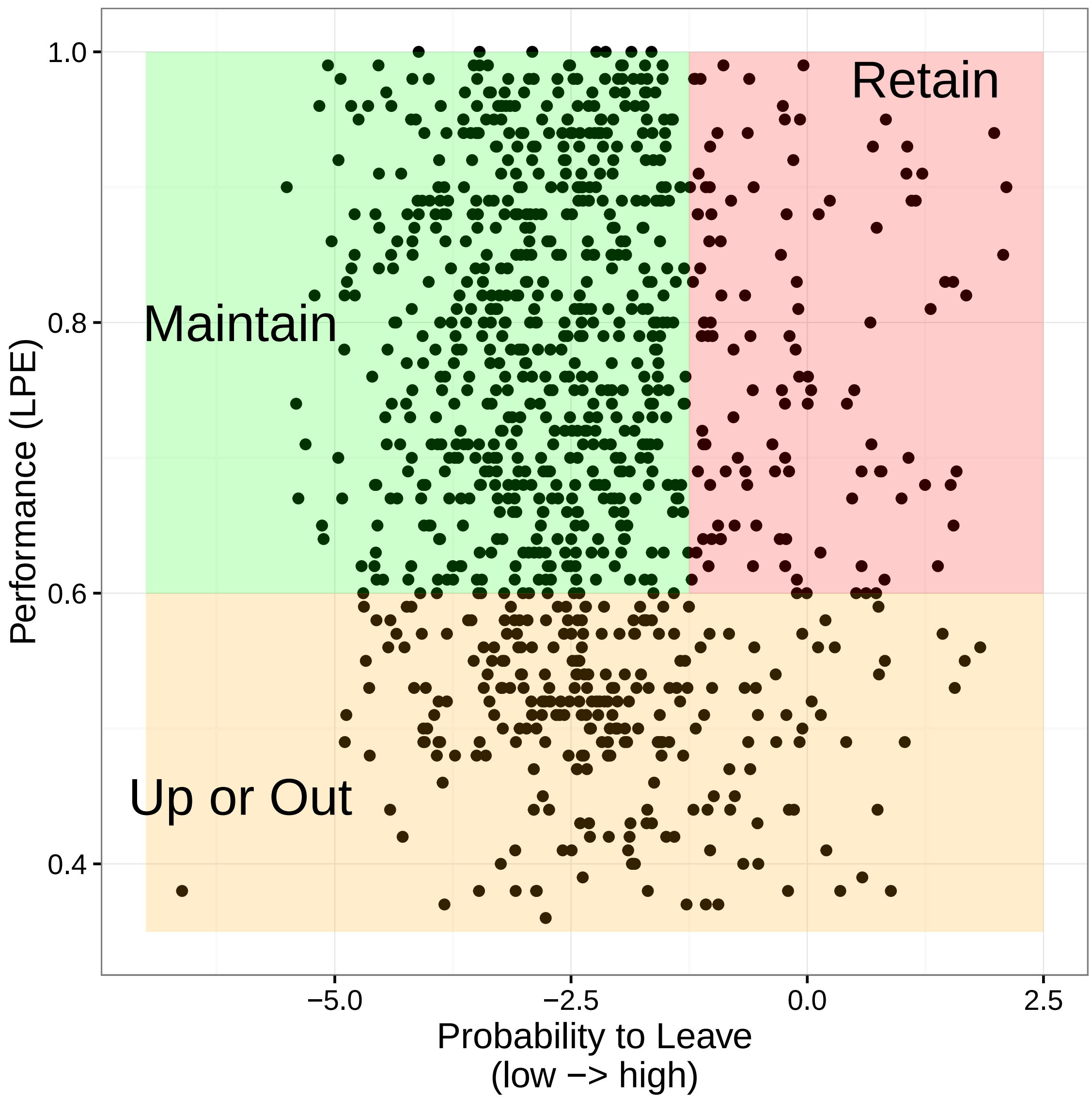
```r
## Fit logistic regressio model
logmod <- glm(left ~ ., family = binomial(logit), data = hr2)

hr3 <- hr3 %>%
  mutate(probaToLeave = predict(logmod, newdata = hr3),
         performance = LPE)

## Prioritize who to try to retain (high performance, high probability to leave)
hr3 <- hr3 %>%
  tbl_df() %>%
  mutate(priority = performance * probaToLeave) %>%
  arrange(-priority)

## Plot probability to leave vs performance
p3 <- ggplot(data = hr3, aes(x = probaToLeave, y = performance)) +
  geom_point() +
  xlab("Probability to Leave\n(low -> high)") +
  ylab("Performance (LPE)") +
  getBaseTheme()
plot(p3)
```




```
## Add priority to the plot
p4 <- p3 +
  geom_point(aes(color = priority))
plot(p4)
```




```
## Add segments to the plot
p5 <- p3 +
  annotate("rect", xmin = -1.25, xmax = 2.5, ymin = 0.6, ymax = 1,
           fill = "red", alpha = 0.2) +
  annotate("rect", xmin = -7, xmax = -1.25, ymin = 0.6, ymax = 1,
           fill = "green", alpha = 0.2) +
  annotate("rect", xmin = -7, xmax = 2.5, ymin = 0.35, ymax = 0.6,
           fill = "orange", alpha = 0.2) +
  annotate("text", x = -6, y = 0.8, label = "Maintain", size = 7) +
  annotate("text", x = -6, y = 0.45, label = "Up or Out", size = 7) +
  annotate("text", x = 1.25, y = 0.98, label = "Retain", size = 7)
plot(p5)
```



3 Predictive Maintenance

```
mnt <- read.csv("DATA_4.03_MNT.csv") %>%
  mutate_if(is.character, as.factor)
str(mnt)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ lifetime      : int  56 81 60 86 34 30 68 65 23 81 ...
##  $ broken        : int  0 1 0 1 0 0 0 1 0 1 ...
##  $ pressureInd   : num  92.2 72.1 96.3 94.4 97.8 ...
##  $ moistureInd   : num  104.2 103.1 77.8 108.5 99.4 ...
##  $ temperatureInd: num  96.5 87.3 112.2 72 103.8 ...
##  $ team          : Factor w/ 3 levels "TeamA","TeamB",..: 1 3 1 3 2 1 2 2 2 3 ...
##  $ provider      : Factor w/ 4 levels "Provider1","Provider2",..: 4 4 1 2 1 1 2 3 2 4 ...
```

3.1 When Will an Part Fail?

The previous analysis predicted an employees probability of leaving but does not include the dimension of time. Will the employees likely to leave move on tomorrow or in 3 years?

What if we want to predict not just whether something will happen, but when. We can use a survival analysis to predict when something will happen. In this example, we use information on mechanical parts like a pressure index, moisture index, and temperature index to predict how soon the part will break.

```
lmod <- lm(lifetime ~ . -broken, data = mnt)
summary(lmod)
```

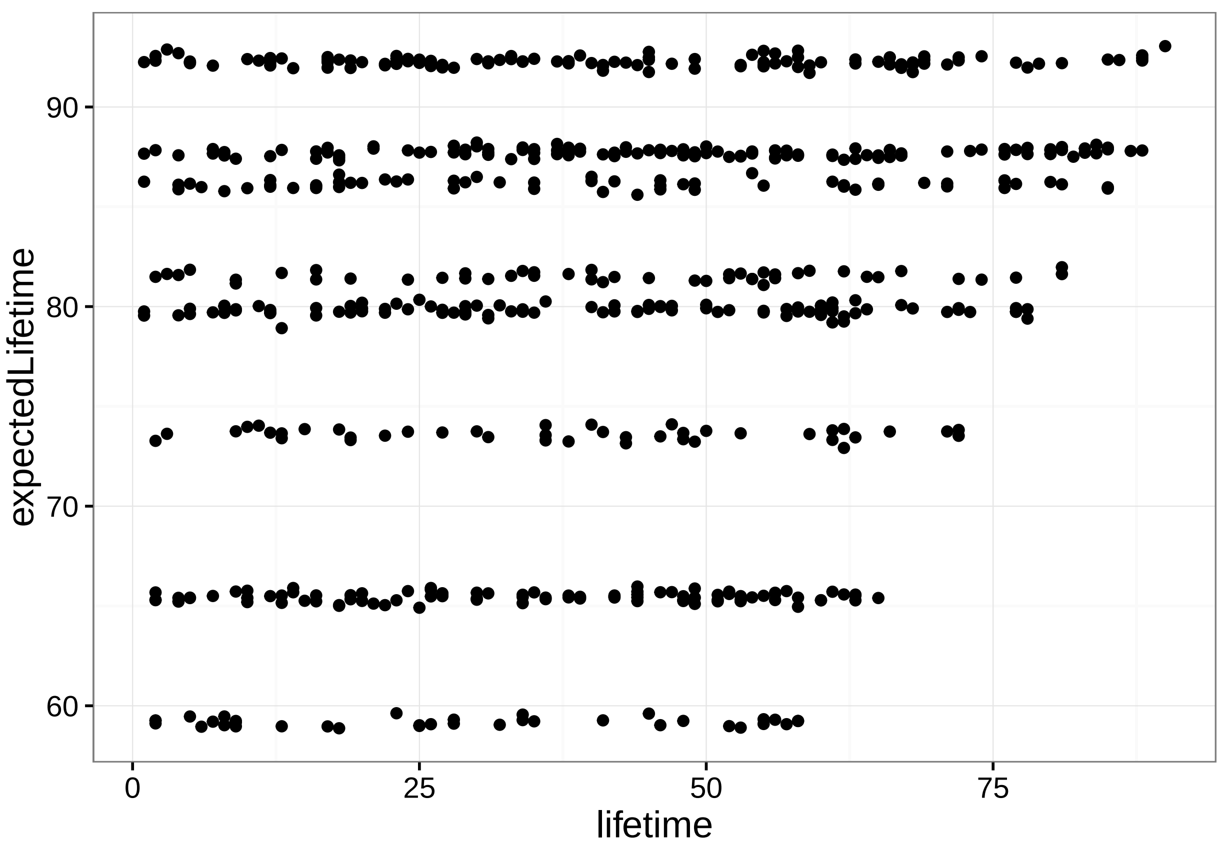
```
##
## Call:
## lm(formula = lifetime ~ . - broken, data = mnt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -59.388 -21.788   8.051  21.112  34.891
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       59.3732039 10.3412622   5.741 1.25e-08 ***
## pressureInd       -0.0031500  0.0416461  -0.076   0.9397
## moistureInd       -0.0173023  0.0830046  -0.208   0.8349
## temperatureInd    -0.0002769  0.0421330  -0.007   0.9948
## teamTeamB          1.5491323  1.9983947   0.775   0.4384
## teamTeamC         -3.4280411  2.0670679  -1.658   0.0976 .
## providerProvider2  0.8835691  2.2944030   0.385   0.7002
## providerProvider3 -9.4858216  2.3490911  -4.038 5.80e-05 ***
## providerProvider4  1.8679357  2.3616268   0.791   0.4292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.13 on 991 degrees of freedom
## Multiple R-squared:  0.0339, Adjusted R-squared:  0.0261
## F-statistic: 4.346 on 8 and 991 DF,  p-value: 3.619e-05
```

```
response <- Surv(time = mnt$lifetime, event = mnt$broken)
survmod <- survreg(response ~ pressureInd + moistureInd +
                     temperatureInd + team + provider,
                   dist = "gaussian", data = mnt)
summary(survmod)
```

```
##
## Call:
## survreg(formula = response ~ pressureInd + moistureInd + temperatureInd +
##     team + provider, data = mnt, dist = "gaussian")
##                      Value Std. Error       z        p
## (Intercept)       8.04e+01    0.29371 273.574 0.00e+00
## pressureInd      -7.14e-04    0.00122  -0.587 5.57e-01
## moistureInd       6.01e-03    0.00240   2.505 1.22e-02
## temperatureInd   -1.04e-02    0.00121  -8.593 8.49e-18
```

```
## teamTeamB          -5.67e-02    0.05882   -0.964 3.35e-01
## teamTeamC          -6.22e+00    0.06132 -101.392 0.00e+00
## providerProvider2  1.25e+01    0.06665  187.464 0.00e+00
## providerProvider3 -1.44e+01    0.06275 -229.241 0.00e+00
## providerProvider4  7.92e+00    0.07056  112.233 0.00e+00
## Log(scale)         -7.43e-01    0.03540  -20.998 6.86e-98
##
## Scale= 0.476
##
## Gaussian distribution
## Loglik(model)= -270.1   Loglik(intercept only)= -1557
##  Chisq= 2573.75 on 8 degrees of freedom, p= 0
## Number of Newton-Raphson Iterations: 12
## n= 1000
```

```r
forecast <- mnt %>%
  mutate(expectedLifetime = predict(survmod, newdata = mnt,
           type = "quantile", p = 0.5)) %>%
  mutate(RemainingLT = expectedLifetime - lifetime) %>%
  filter(broken == 0) %>%
  arrange(RemainingLT)

ggplot(data = forecast) +
  geom_point(aes(x = lifetime, y = expectedLifetime)) +
  getBaseTheme()
```



```
## survfit(response ~ pressureInd + moistureInd + temperatureInd + team + provider,
##         data = mnt)
```

4 Seasonal Sales of Chocolate

```
choc <- read.csv('DATA_4.04_CHOC.csv')
str(choc)
```

```
## 'data.frame':    120 obs. of  4 variables:
##  $ time : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sales: num  135 282 171 171 179 ...
##  $ year : int  2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ month: chr  "01_January" "02_February" "03_March" "04_April" ...
```

```
summary(choc$sales)
```

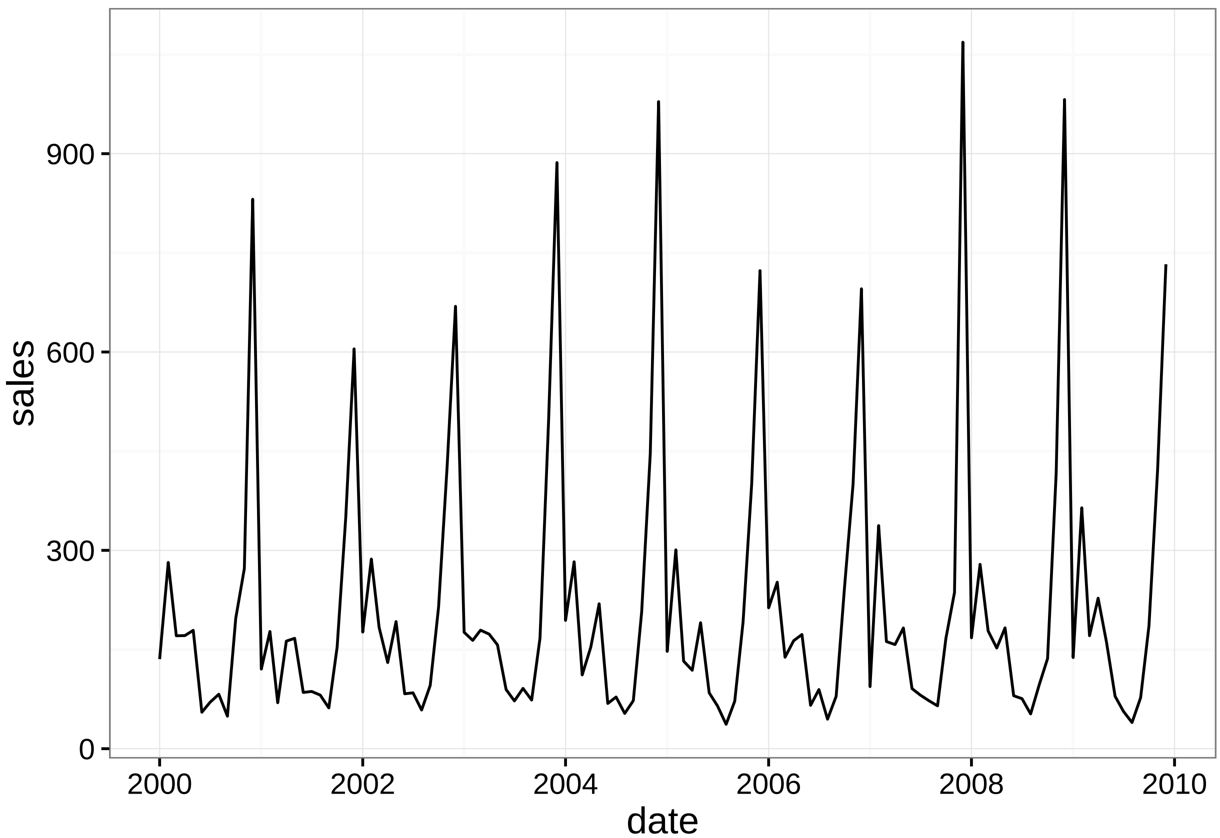
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   36.85   82.88  163.00  216.70  221.30 1069.00
```

```
## Munge the month and year into a date format
## Just assign the first of each month
choc <- choc %>%
    tbl_df %>%
    separate(month, into = c("monthNum", "monthName")) %>%
    mutate(monthName = factor(monthName, levels = month.name)) %>%
    mutate(date = as.Date(paste0("01", monthNum, year), format = "%d%m%Y"))
```

4.1 Visualize Chocolate Sales Over Time

When we visualize chocolate sales over a 10-year span, we notice a seasonal pattern where some parts of the year have very high sales and other parts have very low sales.

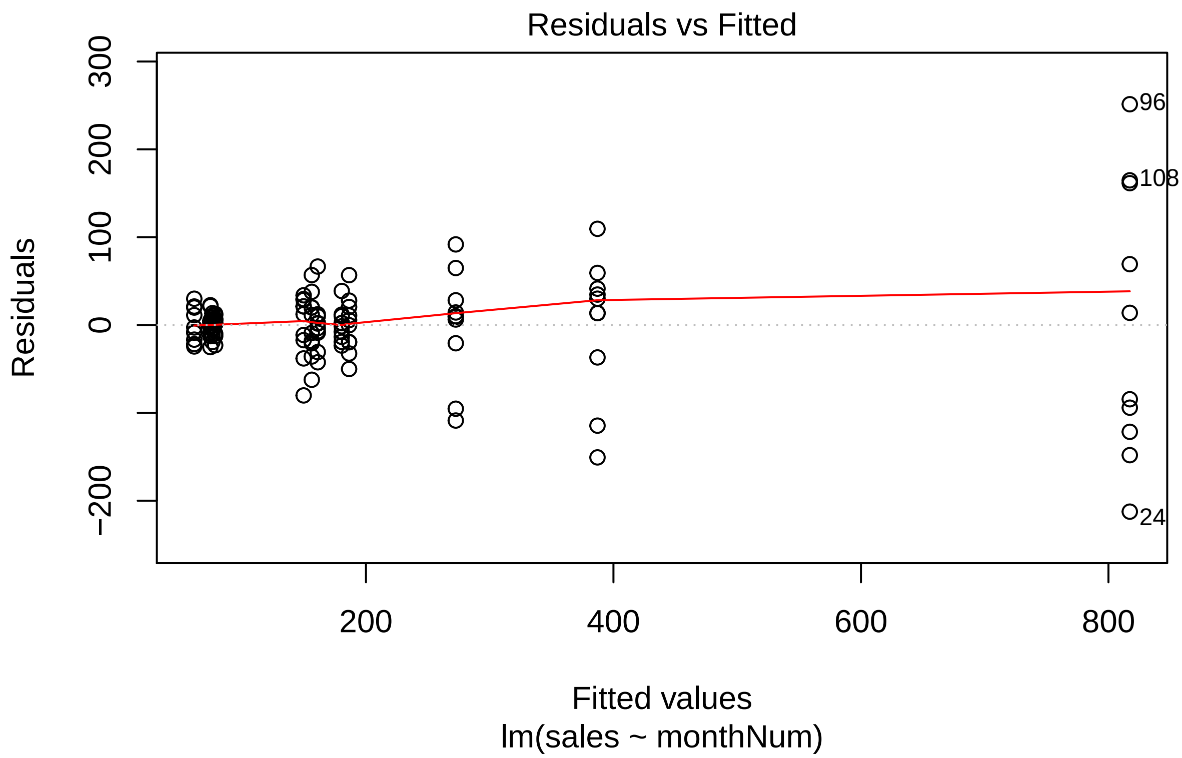
```
p6 <- ggplot(data = choc, aes(x = date, y = sales)) +
    geom_line() +
    getBaseTheme()
plot(p6)
```



4.2 Model Chocolate Sales over Time

We can fit a linear model to predict chocolate sales using the month of the year in order to determine which months are driving this seasonality. Even though the effect of month looks non-linear, this still works well in practice.

```
lmod <- lm(sales ~ monthNum, data = choc)

choc <- mutate(choc, sales_predicted = fitted.values(lmod))

## Residuals vs fitted values.
## The model gets worse at predicting as the sales volume increases
plot(lmod, which = 1)
```




```
summary(lmod)
```

```
##
## Call:
## lm(formula = sales ~ monthNum, data = choc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -212.46  -17.49    2.26   19.87  251.38
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  156.211     18.306   8.533 9.78e-14 ***
## monthNum02   116.377     25.889   4.495 1.75e-05 ***
## monthNum03    -6.559     25.889  -0.253 0.800479
## monthNum04     4.846     25.889   0.187 0.851854
## monthNum05    24.245     25.889   0.937 0.351100
## monthNum06   -78.034     25.889  -3.014 0.003212 **
## monthNum07   -80.262     25.889  -3.100 0.002466 **
## monthNum08   -94.941     25.889  -3.667 0.000382 ***
## monthNum09   -81.921     25.889  -3.164 0.002020 **
## monthNum10    30.185     25.889   1.166 0.246208
## monthNum11   230.894     25.889   8.919 1.33e-14 ***
## monthNum12   661.034     25.889  25.533  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 57.89 on 108 degrees of freedom
## Multiple R-squared:  0.9312, Adjusted R-squared:  0.9242
```

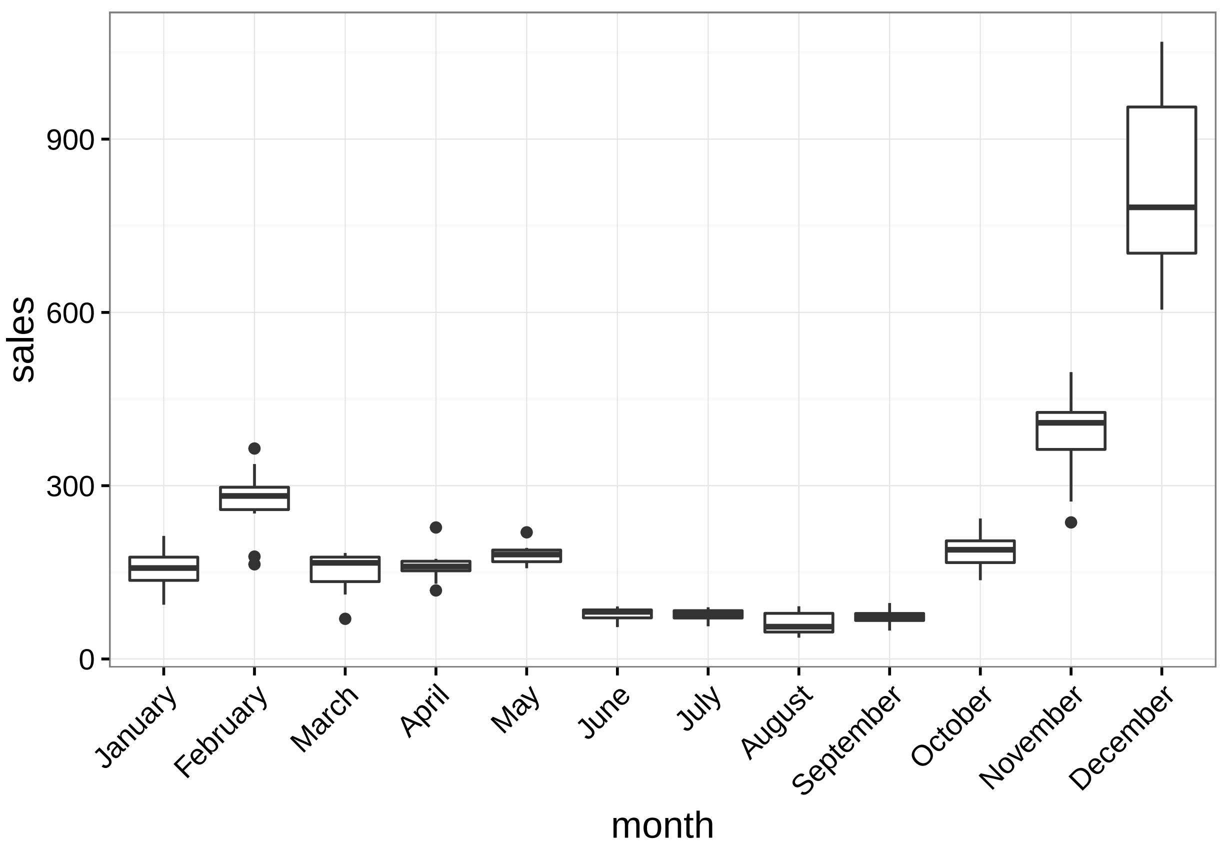
```
## F-statistic: 132.9 on 11 and 108 DF,  p-value: < 2.2e-16
```

We can see from the summary that months 2 (February), 11 (November), and 12 (December) positively predict chocolate sales while summer months 06-09 (June - September) negatively predict chocolate sales.

4.3 Chocolate Sales by Month

Combining all the years together, we can see a boxplot showing the distribution of sales per month.

```
p7 <- ggplot(data = choc, aes(x = monthName, y = sales)) +
    geom_boxplot() +
    xlab("month") +
    getBaseTheme() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
plot(p7)
```

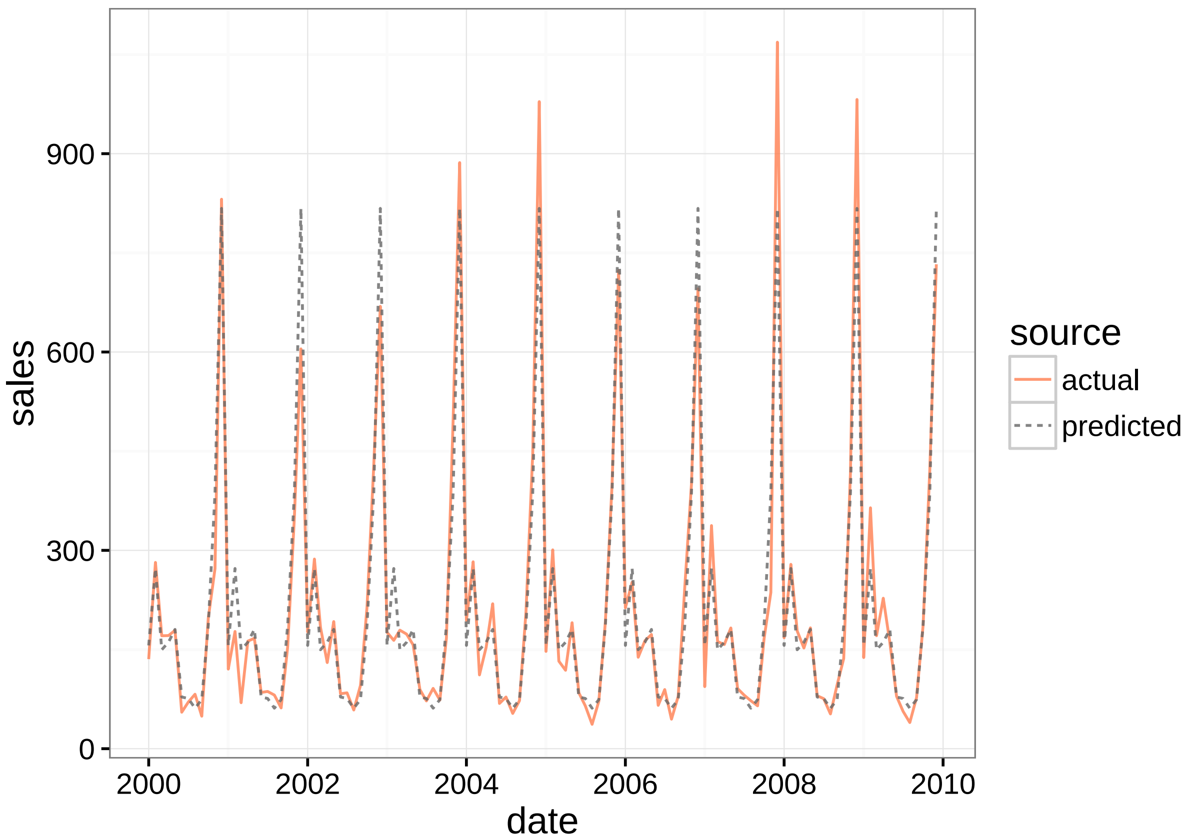


4.4 Recovery Thanks To the Model

```
plotDat <- choc %>%
    rename(sales_actual = sales) %>%
    gather(key = "type", value = "sales", contains("sales_")) %>%
    separate(type, into = c("metric", "source"))
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```r
p8 <- ggplot(data = plotDat, aes(x = date, y = sales,
                                  color = source, lty = source)) +
    geom_line(alpha = 0.8) +
    scale_color_manual(values = c(actual = "coral", predicted = "grey40")) +
    getBaseTheme()
plot(p8)
```



Notice that the predicted value in this simple model does not know anything about the order of the years, it simply predicts the same values every year that on average fit the montly data the best.